AMUN – Autonomic Middleware for Ubiquitious eNvironments Applied to the Smart Doorplate Project

Wolfgang Trumler, Jan Petzold, Faruk Bagci, Theo Ungerer
University of Augsburg
Institute of Computer Science
Eichleitnerstr. 30, 86159 Augsburg, Germany
{trumler,petzold,bagci,ungerer}@informatik.uni-augsbrug.de

March 10, 2004

Abstract

We envision future office buildings that partly or fully implement a flexible office organization. These organizational principles save office space, but require a sophisticated software system that is highly dynamic, scalable, context-aware, self-configuring, self-optimizing and self-healing. We therefore propose an autonomic middleware approach for ubiquitous in-door environments.

1 Introduction

The autonomic computing principles as proposed by IBM in [1] state that future systems should satisfy the self-mechanism, should be anticipatory and context aware. A distributed system like the Smart Doorplate application [3] should be implemented by a distributed solution which provides better scalability of resources and services than a centralized solution does.

2 Application Requirements

To build applications that can react to environmental changes, a sophisticated monitoring is needed to collect and provide information about the services and resources of the nodes and the complete system itself.

To minimize the messaging overhead of a centralized monitor it should be implemented locally for each node. The information exchange for a global view of the monitoring should induce as less communication overhead as possible. The collected information must be analyzed and metrics should calculate a value of excellence. So the metrics may calculate how good a node can offer a special resource or service.

The middleware must support the configuration and reconfiguration of nodes to host new services and to transfer running services to another node. The self-configuration for the initial configuration of the system, self-healing in case of failure and self-optimization during runtime (reconfiguration) are based on this capability and should be fostered by the middleware.

3 AMUN Architecture

AMUN is designed with the goal in mind to foster the device independent application of autonomic computing demands in ubiquitous environments where we expect a heterogeneous collection of devices with diverse capabilities of computing power, memory space, and energy supply. AMUN currently focuses on self-configuration, self-healing and self-optimization as these three mechanisms are base on the configuration and reconfiguration capability of a single node. A key point in the research of AMUN is the usage of simplified mechanisms to find good solutions with an adequate effort to overcome the mostly expensive calculation of optimal solutions.

Proceedings of the International Conference on Autonomic Computing (ICAC'04)
0-7695-2114-2/04 $20.00 © 2004 IEEE

3.1 Middleware Components

AMUN consists of four main parts. The Transport Interface, the Event Dispatcher, the Service Interface and Service Proxy, and the Autonomic Manager.

Transport Interface: The Transport Interface decouples the message delivery from the used transport system. It transforms or encapsulates the EventMessage-Objects to the format of the used transport system. In the current implementation we use JXTA [2].

Event Dispatcher: The Event Dispatcher is responsible for the delivery of incoming and outgoing messages. It offers services the functionality to send messages and to register themselves as listeners to specified types of messages. A service can register for different types of messages and get informed in case of an incoming message with one of the types it registered for.

The Event Dispatcher handles the delivery of broadcast and unicast messages. It knows whether a message can be delivered locally or if the message must be sent to a remote node.

Service Interface and Service Proxy: The Service Interface is the connector between AMUN and a service. A service that wants to participate in AMUN must implement the Service Interface such that it can receive messages delivered by the Event Dispatcher. The interface also offers the functionality for sending messages.

We differentiate between two kinds of Service Interfaces. The simple Service Interface has the full functionality needed to communicate within the system. A Relocatable Service can be transferred to another node whereas a normal service is bound to the node it was started on. The binding on a special node is important for services that need a special hard- or software environment (e.g. fixed sensors or databases).

A Service Proxy is used to forward incoming messages to the new location of a service that has moved to another node. It has a limited lifetime and automatically dies after that time.

Autonomic Manager: The Autonomic Manager is the major control instance of an AMUN node. All relevant information from the local monitors and services are collected here. Autonomic Managers from different nodes exchange information about their state and trigger a reconfiguration if necessary.

The Autonomic Manager contains the Configurator which is responsible for the configuration of the services (init, start, stop, resume, end) on that node. It includes a Control Algorithm which evaluates the "actual state" of the node with the metrics from the Metrics Pool and decides whether a reconfiguration (relocation of a service) should be initiated. The Service Info Space and the Monitor Info Space represent the state repositories for the services respectively for the monitors.

The Control Algorithm and the Metrics are just interfaces in AMUN which must be implemented to adopt AMUN to different special application areas.

4 Conclusion and Future Work

The Autonomic Middleware for Ubiquitous eNvironments (AMUN) facilitates the appliance of self-configuration, self-optimization and self-healing for distributed and service-based ubiquitous applications like the Smart Doorplate application. The distinct monitoring across all layers of the middleware enables AMUN to disburden the administrator from the often complex configuration and optimization of distributed applications.

The next step will be to evaluate the quality and speed of the reconfiguration process. This is vital as it directly affects the self-optimization of the middleware.

References

[1] P. Horn. Autonomic Computing: IBM's Perspective on the State of Information Technology. http://www.research.ibm.com/autonomic/, October 2001.

[2] Project JXTA. http://www.jxta.org, November 2002.

[3] W. Trumler, F. Bagci, J. Petzold, and T. Ungerer. Smart Doorplate - Toward an Autonomic Computing System. In *The Fifth Annual Internatioanl Worshop on Active Middleware Services (AMS2003)*, pages 42–47, Seattle USA, 25. June 2003.